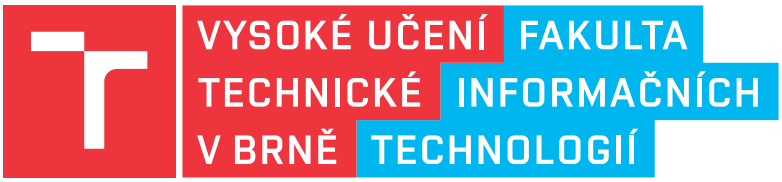


Projektová dokumentace

Překladač jazyka IFJ22

Tým xstrel03
Varianta TRP

6. prosince 2022

Matyáš Strelec	**(xstrel03)**	X%
Ondřej Seidl	(xseidl06)	X%
Maxmilián Nový	(xnovym00)	X%
Dominik Klon	(xklond00)	X%

Obsah

1 Práce v týmu

Rozdělení práce mezi členy týmu (uveďte kdo a jak se podílel na jednotlivých částech projektu; povinně zdůvodněte odchylky od rovnoměrného rozdělení bodů).

1.1 Rozdělení práce

Matyáš Strelec

- Lexikální analýza
- Syntaktická analýzy
- Dokumentace

Ondřej Seidl

- Implementace tabulky symbolů
- Zpracování výrazů

Maxmilián Nový

- Návrh LL-gramatiky
- Vestavěné funkce

Dominik Klon

- Generování kódu

1.2 Odchylky od rozvnoměrného rozdělení

2 Lexikální analýza

2.1 Datové struktury

Implementace lexikální analýzy je obsažena v souborech `lexer.c` a `lexer.h`. Pro potřeby lexikálního analyzátoru byly vytvořeny datové struktury které pomáhají při práci s tokeny a konečným automatem. Výčtový typ `fsm_state_t` obsahuje všechny možné stavy konečného automatu dle návrhu, výčtový typ `token_type_t` definuje typy tokenů.

Struktura `token_t` obsahuje informace o tokenu, jeho typ, pozici v souboru, délku, a jeho předchůdce a následníka ve spojovém seznamu. Struktura `token_list_t` obsahuje ukazatele na první, poslední, a aktuální token.

2.2 Funkce

Všechny funkce jsou ve zdrojových souborech popsané v komentářích, včetně jejich funkcionality, parametrů a návratových hodnot.

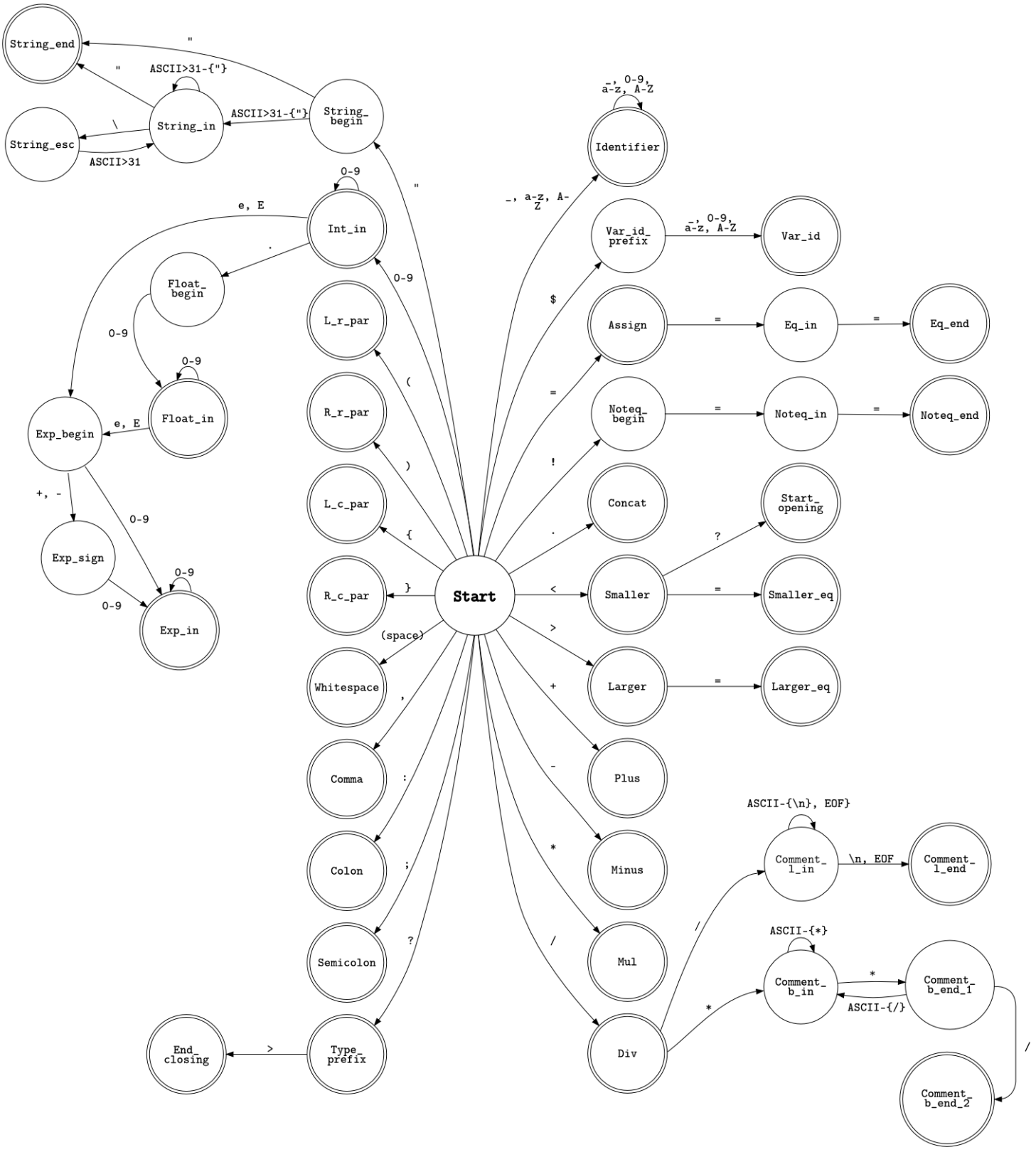
Funkce lexeru volaná z hlavního programu je funkce `fillTokenList()`, která jako parametr dostává ukazatel na strukturu `token_list_t`, kterou naplní seznamem tokenů pomocí volání funkce `getNextToken()`. Funkce `getNextToken()` je volána v cyklu, dokud není dosažen token typu konec souboru.

Funkce `getNextToken()` je implementována pomocí konečného automatu. Dle posloupnosti znaků na vstupu určuje typ a vyplňuje data tokenu. V případě, že je na vstupu znak, který nelze podle automatu dále číst, je kontrolováno, jestli momentální stav automatu je koncový, pokud ano, token je validní. Dále jsou rozpoznána klíčová slova a odstraněny úvozovky z řetězců. Pokud automat není v koncovém stavu, ale na vstup přijde znak, který automat nemůže přečíst, funkce vrací chybu 1.

Dále soubor obsahuje funkce na práci se seznamem tokenů jako vázaným seznamem a funkce pro ladění.

2.3 Diagram konečného automatu

Vizte obrázek 1.



Obrázek 1: Diagram konečného automatu vytvořený nástrojem `Graphviz`

3 Syntaktická analýza

3.1 Implementace

3.2 LL-gramatika

Pro jazyk IFJ22 byla navržena následující gramatika.

```
1:  <prog> -> <stat> <prog>
2:  <prog> -> function func-id ( <params> ) : type { <st-list> } <prog>
3:  <prog> -> <eof>

4:  <eof> -> ?> EOF
5:  <eof> -> EOF

6: <params-cont> -> , type $id <params-cont>
7: <params-cont> -> eps

8: <params> -> type $id <params-cont>
9: <params> -> eps

10: <args-cont> -> , <term> <args-cont>
11: <args-cont> ->

12: <args> -> <term> <args-cont>
13: <args> -> eps

14: <stat> -> $id = <assign> ;
15: <stat> -> while ( <expr> ) { <st-list> }
16: <stat> -> if ( <expr> ) { <st-list> } else { <st-list> }
17: <stat> -> return <expr> ;
18: <stat> -> <expr> ;
19: <stat> -> func-id ( <args> ) ;

20: <st-list> -> <stat> <st-list>
21: <st-list> -> eps

22: <assign> -> <expr>
23: <assign> -> func-id ( <args> )

24: <term> -> $id
25: <term> -> val
```

Poznámky

`$id` - identifikátor proměnné
`func-id` - identifikátor funkce
`val` - číselný nebo řetězcový literál
`type` - datový typ (`int`, `double`, `string`)
`func-id` - identifikátor funkce
`eps` - ε

3.3 LL-tabulka

	function	func-id	(	)	:	type	{	}	?>	EOF	,	$id	=	;	while	<expr>	if	else	return	val
<prog>	2	1							3	3		1			1	1	1		1	
<eof>									4	5										
<params-cont>				7							6									
<params>				9		8														
<args-cont>				11							10									
<args>				13								12								12
<stat>		19										14			15	18	16		17	
<st-list>		20						21				20			20	20	20		20	
<assign>		23														22				
<term>												24								25

3.4 Precedenční tabulka

	*	/	+	-	.	<	>	<=	>=	===	!==	(	)	var	$
*	>	>	>	>	>	>	>	>	>	>	>	<	>	<	>
/	>	>	>	>	>	>	>	>	>	>	>	<	>	<	>
+	<	<	>	>	>	>	>	>	>	>	>	<	>	<	>
-	<	<	>	>	>	>	>	>	>	>	>	<	>	<	>
.	<	<	>	>	>	>	>	>	>	>	>	<	>	<	>
<	<	<	<	<	<	>	>	>	>	>	>	<	>	<	>
>	<	<	<	<	<	>	>	>	>	>	>	<	>	<	>
<=	<	<	<	<	<	>	>	>	>	>	>	<	>	<	>
>=	<	<	<	<	<	>	>	>	>	>	>	<	>	<	>
===	<	<	<	<	<	<	<	<	<	>	>	<	>	<	>
!==	<	<	<	<	<	<	<	<	<	>	>	<	>	<	>
(	<	<	<	<	<	<	<	<	<	<	<	<	=	<	
)	>	>	>	>	>	>	>	>	>	>	>		>		>
var	>	>	>	>	>	>	>	>	>	>	>		>		>
$	<	<	<	<	<	<	<	<	<	<	<	<		<	

4 Sémantická analýza

5 Tabulka symbolů

6 Generování kódu

7 Hlavní program